EXPERIMENT - 12

Object Oriented Programming Lab

Aim

Write a program to perform addition of two complex numbers using constructor overloading. The first constructor which takes no argument is used to create objects which are not initialized, second which takes one argument is used to initialize real and image parts to equal values and third which takes two argument is used to initialized real and image to two different values.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 12

Aim:

Write a program to perform addition of two complex numbers using constructor overloading. The first constructor which takes no argument is used to create objects which are not initialized, second which takes one argument is used to initialize real and image parts to equal values and third which takes two argument is used to initialized real and image to two different values.

Theory:

A class **constructor** is a special member function of a class that is executed whenever we create new objects of that class.

A constructor will have exact same name as the class and it does not have any return type at all, not even void. Constructors can be very useful for setting initial values for certain member variables.

Parameterized Constructor

A default constructor does not have any parameter, but if you need, a constructor can have parameters. This helps you to assign initial value to an object at the time of its creation.

Copy Constructor

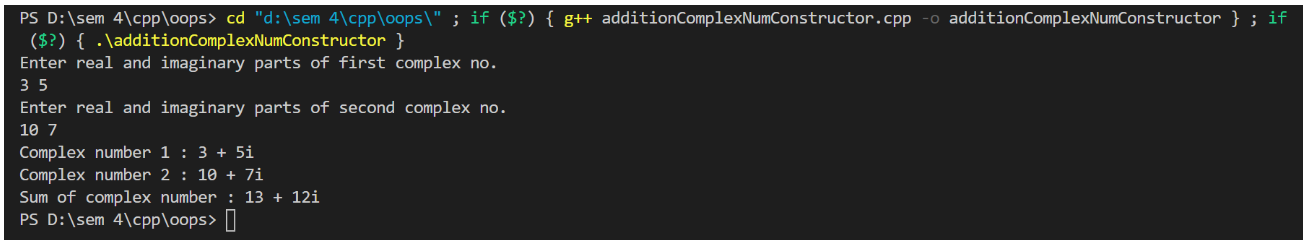
The copy constructor is a constructor which creates an object by initializing it with an object of the same class, which has been created previously.

Source Code:

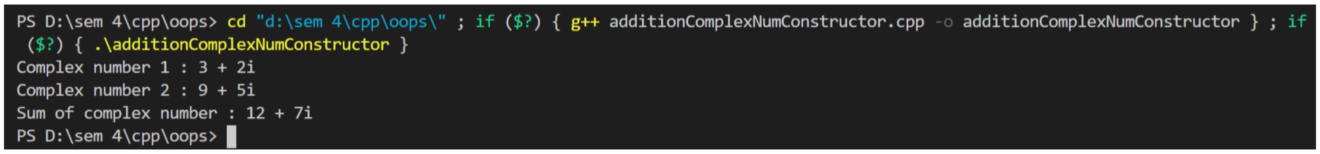
```
#include <iostream>
using namespace std;

class Complex{
    public:
        int real, imaginary; //declaration of variables to be used

        // Empty Constructor
        Complex(){}
        // Constructor to accept real and imaginary part
        Complex(int r, int i){
            real = r;
            imaginary = i;
        }

        // func for adding two complex number
        Complex addComp(Complex C1, Complex C2){
            Complex res;
            res.real = C1.real + C2.real;
            res.imaginary = C1.imaginary + C2.imaginary;
            return res;
        }
};

int main(){
    // // complex num 1
    // Complex C1(3, 2);
    // cout << "Complex number 1 : " << C1.real << " + " << C1.imaginary << "i" <
<endl;

    // // complex Num 2
    // Complex C2(9, 5);
    // cout << "Complex number 2 : " << C2.real << " + " << C2.imaginary << "i" <
< endl;

    // Complex C3;
    // C3 = C3.addComp(C1, C2);

    // cout << "Sum of complex number : " << C3.real << " + " << C3.imaginary <<
"i" << endl;

    int c1, r1, c2, r2;
    cout << "Enter real and imaginary parts of first complex no. " << endl;
    cin >> c1 >> r1;
    cout << "Enter real and imaginary parts of second complex no. " << endl;
    cin >> c2 >> r2;
    Complex C1(c1, r1);
    cout << "Complex number 1 : " << C1.real << " + " << C1.imaginary << "i" <<en
dl;

    // complex Num 2
    Complex C2(c2, r2);
    cout << "Complex number 2 : " << C2.real << " + " << C2.imaginary << "i" << e
ndl;
```

```
    Complex C3;
    C3 = C3.addComp(C1, C2);

    cout << "Sum of complex number : " << C3.real << " + " << C3.imaginary << "i"
 << endl;

}
```

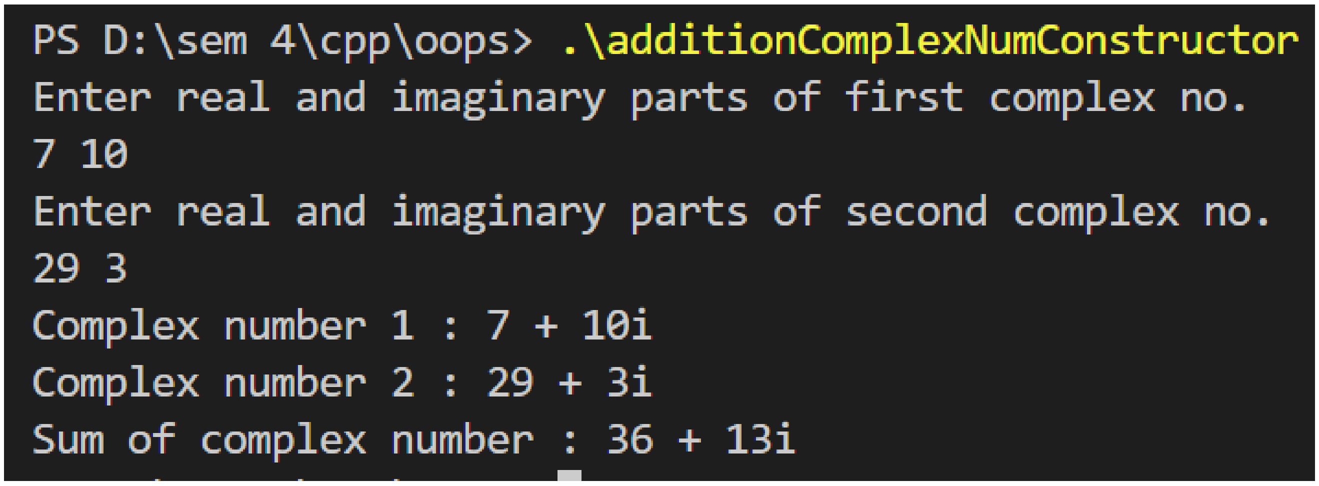
Output:



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ additionComplexNumConstructor.cpp -o additionComplexNumConstructor } ; if
 ($?) { .\additionComplexNumConstructor }
Enter real and imaginary parts of first complex no.
3 5
Enter real and imaginary parts of second complex no.
10 7
Complex number 1 : 3 + 5i
Complex number 2 : 10 + 7i
Sum of complex number : 13 + 12i
PS D:\sem 4\cpp\oops> 
```



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ additionComplexNumConstructor.cpp -o additionComplexNumConstructor } ; if
 ($?) { .\additionComplexNumConstructor }
Complex number 1 : 3 + 2i
Complex number 2 : 9 + 5i
Sum of complex number : 12 + 7i
PS D:\sem 4\cpp\oops> 
```



```
PS D:\sem 4\cpp\oops> .\additionComplexNumConstructor
Enter real and imaginary parts of first complex no.
7 10
Enter real and imaginary parts of second complex no.
29 3
Complex number 1 : 7 + 10i
Complex number 2 : 29 + 3i
Sum of complex number : 36 + 13i
```

Viva Questions

Q1) What is a constructor?

Ans.

A **constructor** is a function of a class that has the same name as the class. The constructor is called at the time of the initialization of object. There are three types of constructors –

- Default constructor
- Parameterized constructor
- Copy constructor

Syntax

```
class cl_name{
        cl_name(){
        //This is constructor..
        }
}
```

Q2) What is a destructor?

Ans.

A **destructor** is a method of a class that has the same name as the class preceded by a **tild ~** symbol. It is called at the end of code or when the object is destroyed or goes out of scope.

Syntax

```
class cl_name{
        ~ cl_name(){} //destructor
}
```

Q3) What is the use of constructor?

Ans.

A constructor is a method that has the same name as a class. And the use of a constructor is to initialize the object when it is created using a **new** keyword.

When an object is created, the variables are initialized chunks of memory and base values if there are any.

Q4) What is the use of destructor?

Ans.

A destructor is a method that has the same name as a class preceding a ~ symbol. The use of a destructor is to deallocate the memory chunks one the code goes out of the scope of the object or deleted using the **delete keyword**.

When the object is deleted the destructor is called and it deallocated all the memory blocks that were created when an object was created.

Q5) What is the order of constructor execution in C++?

Ans.

A constructor is invoked when the object of a class is created. The order in which a constructor is invoked is the same as the hierarchy of the inheritance. This means that first the object of a base class is invoked then the objects of the child class are invoked and so on.

Q6) What is the order of destructor execution in C++?

Ans.

A destructor is invoked in the reverse order as the constructor and is invoked when the object of the class is deleted. The order in which a destructor is invoked is just the opposite of the hierarchy of the inheritance. This means that first the object of child class is destroyed then the objects of the parent class are destroyed and so on.

Q7) Is the default constructor created even if we create any other constructor?

Ans.

Constructors are created by default by the compiler if a programmer doesn't define any constructor explicitly. If the programmer defines a constructor then compiler holds its work and does not define any of it.